2009 International Conference on Computer Engineering and Applications
IPCSIT vol.2 (2011) © (2011) IACSIT Press, Singapore

Network Intrusion Alert Aggregation Based on PCA and Expectation Maximization Clustering Algorithm

Maheyzah Md Siraj [+], Mohd Aizaini Maarof and Siti Zaiton Mohd Hashim

Faculty of Computer Science and Information Systems
Universiti Teknologi Malaysia
81310 Skudai Johor, Malaysia

Abstract. Most of the organizations implemented various security sensors for increased information security and assurance. A popular choice is Network Intrusion Detection Systems (NIDSs). Unfortunately, NIDSs trigger a massive amount of alerts even for a day and overwhelmed security experts. Worse, a large number of these alerts are false positives, and redundant warnings for the same attack, or alert notifications from erroneous activity. Such low quality of alerts gives negative impact to the alert analysis. We propose an alert aggregation model based on Principal Component Analysis (PCA) coupled with unsupervised learning clustering algorithm - Expectation Maximization for Gaussian Mixture (EM_GM) to aggregate similar alerts and to reduce the number of alerts. Our empirical results show that the proposed model effectively clustered NIDSs alerts and significantly reduced the alert volume.

Keywords: alert clustering, alert filtering, alert aggregation, alert reduction, PCA, EM_GM

1. Introduction

Network Intrusion Detection Systems (NIDSs) have been extensively used by researchers and practitioners to monitor intrusive activities in computer networks. NIDSs usually generated thousands of alerts even for a day. Worse, those alerts are mixed with false positives, and repeated warnings for the same attack, or alert notifications from erroneous activity [1]. Therefore, manually analyze those alerts are tedious, time-consuming and error-prone [1].

A promising technique to automatically analyze the intrusion alerts is called *correlation*. Alert Correlation Systems (ACS) is post-processing modules that provide high-level insight on the security state of the network and filter false positives as well as redundant alerts efficiently from the output of NIDSs [2]. The analysis results actually become an important guidance for the security expert (SE) to plan and develop the responsive and preventive mechanisms. Generally, correlation can be of two types: *structural* correlation and *causal* correlation. In this paper, we address the structural correlation (or alert clustering) aspect of NIDSs data to group (or aggregate) alerts with similar features.

The main problem in existing ACSs is they require high levels of human SE involvement in creating the system and/or maintaining it. For instance, algorithm introduced by [3] required a significant amount of alerts to be managed manually (i.e., hand-clustered) beforehand. Likewise, system by [1], it required manual tuning periodically. Moreover, in their first system deployment, it needs to encode network properties to assist the clustering algorithm. These approaches were time-consuming since regular setup and maintenance are significantly required for their system. Therefore, those constraints make the development of supervised learning-based correlation system less practical. Our goal is to minimize the intervention (i.e., to ease the burden) of SE as much as possible, but not to replace them. Therefore, an unsupervised learning-based

[+] Corresponding author. Tel.: +607 5532245; fax: +607 5593185.
E-mail address: maheyzah@utm.my

clustering model is proposed to reduce the number of alerts and to discover the attack steps launched by attackers. We propose a new hybrid method called Principal Component Analysis (PCA) and Expectation Maximization for Gaussian Mixture (EM_GM), for alert aggregation and filtration. The closest work to ours was by [6] which used the Expectation Maximization clustering algorithm as well but a major different is that we implemented PCA to obtain better performance.

The following section outlines our proposed architecture, with detail elaboration for each component involved in the architecture. Section 3 explains the dataset, the experiments and discussions of the results obtained. Lastly, we conclude the paper and present our potential future work.

2. System Architecture

The goal of this work is to find the best couple of PCA and unsupervised learning clustering algorithm for alert aggregation and filtration. Our general architecture consists of six components as visualized in Fig. 1 (i.e., alert normalization, preprocessing, dimension reduction using PCA, alert aggregation/clustering using unsupervised learning algorithms, alert ranking and verification, and alert reduction).

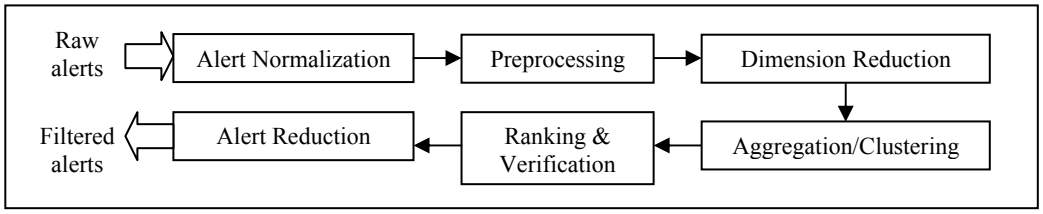

Fig. 1: General architecture for alert aggregation and filtration in Alert Correlation System.

2.1. Alert Normalization

Recently, organizations used cooperative NIDSs to provide a better detection and global view of intrusion activities. This contributes to the diversity of output formats. In order to correlate alerts such diversified formats have to be converted into a unified standard representation. We applied IDMEF [4] to define the common data formats for the alerts. We extracted nine attributes for each alert. Thus, a vector for an alert A = {*AlertID*, *SensorID, DetectTime, SourceIPAddress, SourcePort, DestinationIPAddress, DestinationPort, ServiceProtocol, AlertType*}. All alerts are stored in a relational database.

2.2. Preprocessing

Alert attributes are in the form of numerical and non-numerical values. Attributes that contain numerical values are *AlertID, SensorID, SourcePort*, *DestinationPort*, and *DetectTime*. The rest are non-numerical (i.e., *SourceIPaddress*, *DestinationIPaddress, ServiceProtocol* and *AlertType*), and have to be mapped into a numerical value. For instance to convert a 32-bit IP address ($X1.X2.X3.X4$), this mapping is used: IP_{num} = (($X1$ x 256 + $X2$) x 256 + $X3$) x 256 + $X4$. We scaled all values in the range of [0,1] using Improved Unit Range (IUR) scaling method as in (1) because it eliminates zero values.

$$x' = 0.8 \text{ x } \frac{(x - x_{min})}{(x_{max} - x_{min})} + 0.1 \quad (1)$$

where x' is the scaled value, x is raw value, x_{max} is maximum value and x_{min} is minimum value.

2.3. Dimension Reduction using PCA

PCA has proven to be a useful technique for dimension reduction and multivariate analysis. An important virtue of PCA is that the extracted components are statistically orthogonal to each other. This contributes to speed training and robust convergence. We expect that the unsupervised learning algorithm can work much better with PCA. According to [5], the definition of PCA is that, for a set of observed vectors $\{v_i\}$, $i \in \{1,2,...,N\}$, the q principle axes $\{w_j\}$, $j \in \{1,2,..,q\}$ are those orthonormal axes onto which the retained variance under projection is maximal. It can be shown that the vectors w_j are given by the q dominant eigenvectors (i.e. those with largest associated eigenvalues) of the covariance matrix

$C = \sum_i \frac{(v_i - \bar{v})(v_i - \bar{v})^T}{N}$ such that $Cw_j = \lambda_i w_j$, where $\bar{v}$ is the simple mean. The vector $u_i = W^T(v_i - \bar{v})$, where $W = (w_1, w_2, ..., w_q)$, is thus a q-dimensional reduced representation of the observed vector $\{v_i\}$.

For the intrusion alerts in the dataset, the purpose of performing PCA is to find the principal components of the alerts (i.e., the attributes vector that can describe the alerts exactly and sufficiently, but not redundantly). In mathematical terms, we wish to find the principal components of the distribution of the alerts, or the eigenvectors of the covariance matrix of the set of the alerts [5].

2.4. Alert Aggregation/Clustering using Unsupervised Learning Algorithms

We tested two unsupervised learning clustering algorithms: Self-Organizing Maps (SOM) and EM_GM, for performance comparison. Noted that, in this clustering component, we did not include the *AlertID*, *SourceIPAddress,* and *DestinationIPAddress* attributes because as mentioned in [6], IP address tended to impede correct clustering since they are easily forged. However these attributes will be used for the next stage of correlation in our future work.

SOM [7] is a competitive learning algorithm that reduces the dimensions of data by mapping high dimensional data onto a set of units set up in a 2-dimensional lattice. An n-dimensional weight vector is associated with each unit, having the same dimension of the input space. At each step, the Euclidean distances between a randomly selected input vector *x* and all the units weight vectors w_i is calculated. The unit having the shortest distance to the input vector is identified to be the best matching unit *c* for *x.* As a result, the winner index *c* or best matching unit (BMU) for input vector *x(t)* is identified. Then, the input is mapped to the location of the BMU. We updated the weight vectors of the units neighbouring the BMU(*c)* and of BMU itself, $w_i(t+1) = w_i(t) + \delta(t)[(x(t) - w_i(t))]$; for $i = c$ and its neighbours.

On the other hand, the EM_GM algorithm [8] consists of two repeated steps, *Expectation* and *Maximization* (EM). It uses a statistical model called *Gaussian finite mixtures* (GMM) to achieve the goal of producing the most likely set of clusters given the number of clusters, *k*, and a set of data. The model consists of a set of *k* probability distributions, one to represent the data of each cluster. There are parameters (e.g, *number of iteration* and *log likelihood difference between two iterations*) that define each of the *k* distributions. The EM_GM algorithm begins by making initial guesses for these parameters based on the input data, then determines the probability that a particular data instance belongs to a particular cluster for all data using these parameter guesses. The distribution parameters are revised again and this process is repeated until the resulting clusters have some level of overall cluster '*goodness*' or until a maximum number of algorithm iterations are reached.

In particular, it attempts to find the parameters θ that maximize the log probability log$P(x;\theta)$ of the observed data. It reduces the difficult task of optimizing log$P(x;\theta)$ into a sequence of simpler optimization subproblems, whose objective functions have unique global maxima that can often be computed in closed form. These subproblems are chosen in a way that guarantees their corresponding solutions $\phi^{(1)}$ $\phi^{(2)}$,... and will converge to a local optimum of log$P(x;\theta)$.

More specifically, the *Expectation* step (E-step) of the algorithm estimates the clusters of each data instance given the parameters of the finite mixture [6]. During the E-step, the algorithm chooses a function g_t that lower bounds log$P(x;\theta)$ everywhere, and for which $g_t(\phi^{(t)})$=log$P(x; \phi^{(t)})$. The *Maximization* step (M-step) of the algorithm tries to maximize the likelihood of the distributions that make up the finite mixture, given the data [6]. During the M-step, the algorithm moves to a new parameter set $\phi^{(t+1)}$, that maximizes g_t. As the value of the lower-bound g_t matches the objective function at $\phi^{(t)}$, it follows that:

$$\log P(x; \phi^{(t)}) = g_t(\phi^{(t)}) \leq g_t(\phi^{(t+1)}) = \log P(x; \phi^{(t+1)}) \quad (2)$$

so the objective function monotonically increases during each iteration of EM_GM.

2.5. Alert Ranking and Verification

Alerts that issued by NIDSs are not all on the same level of severity and importance. It would be great if the system can identify which alerts are highly important and which are not, so that the number of alerts that need to be deal with can be reduced. We automatically cross-checked each alerts with the sensor's signatures file [9] to determine the priority of alerts and to verify the false positive and invalid alerts. In alert ranking,

we introduced three level of severity: (1) *High-risk,* (2) *Medium-risk* and (3) *Low-risk*. For each level, we associate a numerical weight of priority in order to distinguish significant alarms from the others.

2.6. Alert Reduction

Given the clustered alerts from previous component, redundant alerts (i.e., equal values in all attributes) in each cluster are merged into a hyper-alert. In specific, repeated alerts for each cluster are represented as one. Moreover, with the reduction of invalid, false positive and low risk alerts, the total number of alerts left for future analysis is significantly reduced.

3. Experiments And Dataset

The experiments were conducted with MIT Lincoln's Lab's DARPA 2000 Scenario Specific Dataset [10]. The dataset contain simulated multi-staged attack scenarios in a protected environment: the intruder probes, breaks-in, installs the Distributed Denial-of-Service (DDoS) daemon and launches a DDoS attack against an off-site server. Since we are dealing with the sensor data, alerts reported by RealSecure network sensor Version 6.0 which are provided by [11] are used to evaluate the effectiveness of our model. The alerts data represents two kinds of attack scenarios (i.e., scenario 1.0 and scenario 2.0.2) in two types of networks (i.e., *inside* and *dmz* network). Attacks in scenario 2.0.2 were stealthier than scenario 1.0. For this paper, we only used alerts data for scenario 2.0.2 in *dmz* network. For implementation of the model, we used MATLAB Software [12]. Since SOM is used in [13] for clustering alerts, we used it as a benchmark to compare the performance of our proposed model. Thus, we have two set of experiments: (1) clustering (i.e., SOM and EM_GM) without PCA, and (2) clustering with PCA.

3.1. Results and Data Analysis

The results for unsupervised learning clustering algorithms are showed in Table I. The number of alerts tested is 430. The results obtained were compared against the benchmark clusters (i.e., 16 clusters are expected) to determine performance. As in Table I, we used four measurements: (1) *Clustering Error* (CE) is the number of alerts that are wrongly clustered. (2) *Error rate* (ER) is the percentage of wrongly clustered alerts, ER = (CE ÷ Total number of alerts observed) x 100, (3) *Clustering Accuracy Rate* (CAR) is the percentage of alerts that are accurately clustered as they should be, CAR = 100 – ER, and (4) *Time* is the algorithm's processing time in seconds.

We tested the SOM by simultaneously varying the epochs and lattice configuration. Two third of the dataset was used for training and the rest is for testing. The best result on SOM (i.e., 73.58% with PCA) is attained after it is trained for 2500 epochs using hexagonal 4 by 6 lattice type. It produced 12 clusters. In term of time costs, the overall processing time for training and testing is 7.42 seconds. The processing time might be longer if the dataset, epochs and/or lattice type are larger.

Similarly, we varied the number of clusters in EM_GM. The best performance for EM_GM (i.e., 90.33% with PCA) is reached at 14 clusters and the processing time is 4.59 seconds. In each cluster, similar types of alerts are grouped together to represent an attack step. Since SOM produced 12 clusters, it means that SOM put a large number of alerts that should belong together in one cluster into another clusters. Therefore, we summarized that the proposed model (i.e., PCA-EM_GM) is effective and performed better than the SOM algorithm for this dataset in term of clustering accuracy and processing time.

TABLE I. CLUSTERING PERFORMANCE

Model	SOM [13]				EM_GM			
	CE	ER (%)	CAR (%)	Time (sec)	CE	ER (%)	CAR (%)	Time (sec)
Without PCA	135	31.84	68.16	4.21	45	10.61	89.39	1.85
PCA	112	26.42	73.58	7.42	**41**	**9.67**	**90.33**	**4.59**

Table II presented the type of alerts with their level of rank/priority. It shows that the majority of the alerts are most probably not serious at all. SE might found such alerts inappropriate to be analyzed and correlated. But, others may feel they are appropriate. Because of that, our system provided two kind of mode to automatically delete the low risk alerts: (1) need permission from SE, or (2) no need. If (2) is chosen, then

the total reduction of alerts is significant. The original input data was 430, thus the percentage of total reduction of unwanted alerts is 87.67% (i.e., 3 merged, 2 invalid, 1 false positive, and 371 low risk alerts).

TABLE II. TOTAL OF RANKED ALERTS

Rank	Type of Alerts	Total
High-risk	Admind, HTTP_ActiveX, HTTP_Cisco_Catalysts_Exec, Sadmind_Amslverify_Overflow	10
Medium-risk	Email_Almail_Overflow, FTP_Pass	43
Low-risk	Email_Ehlo, FTP_Put, FTP_Syst, FTP_User, HTTP_Java, SSH_Detected, TCP_Urgent_Data, TelnetEnvAll, TelnetTerminalType, TelnetXdisplay	371

4. Conclusion And Future Work

We proposed a PCA and EM_GM-based clustering algorithm to aggregate alerts and to reduce the number of alerts. This model is useful for clustering and filtering NIDSs output data which are huge and extremely hard to be analyzed manually. Altogether, the results are encouraging in term of clustering accuracy rate and processing time compared to SOM. In both algorithms, PCA has proven to be an excellent choice for dimension reduction and boost clustering performance.

Noted that a successful network attack consists of multi-stages attack, and an attack stage may comprise of one/more attack steps. Thus, we need another clustering component to aggregate similar attack types to reveal the stages of attack. This will becomes our main future work besides testing the proposed model with larger dataset. In the near future, we would like to experiment with other clustering algorithms and develop an adaptive multi-stages correlation system to determine known and unknown attack scenarios.

5. References

[1] K. Julisch, and M. Dacier. Mining Intrusion Detection Alarms for Actionable Knowledge. *Proc. of the 8th ACM Int. Conf. on Knowledge Discovery and Data Mining*. 2002, pp. 366–375.

[2] F. Valeur, G. Vigna, and C. Kruegel. A Comprehensive Approach to Intrusion Detection Alert Correlation. *IEEE Trans. on Dependable and Secure Computing*. 2004, **1** (3): 146-169.

[3] O.M. Dain, and R. K. Cunningham. Fusing a Heterogeneous Alert Stream into Scenarios. *ACM Workshop on Data Mining for Security Applications*. 2001, pp. 1-13.

[4] H. Debar, D. Curry, and B. Feinstein. The Intrusion Detection Message Exchange Format (IDMEF). ftp://ftp.rfc-editor.org/in-notes/rfc4765.txt, (current Mar. 2007).

[5] I. T. Jolliffe. *Principal Component Analysis (3rd ed.)*. Springer Verlag, 2002.

[6] N. Japkowicz, and R. Smith. *Autocorrel II: Unsupervised Network Event Correlation using Neural Networks*. Contractor Report CR2005-155. DRDC Ottawa, 2005.

[7] T. Kohonen. *Self-Organizing Maps: Series in Information Sciences (3rd ext. ed.)*. Springer, 2001.

[8] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum Likelihood from Incoming Data via the EM Algorithm. *J. Royal Stat. Soc., Series B*. 1977, **39** (1): 1–36.

[9] RealSecure Signatures Reference Guide. Internet Security Systems. http://xforce.iss.net/, (current Jan. 2008).

[10] MIT Lincoln Lab. DARPA 2000 Scenario Specific Datasets. http://ideval.ll.mit.edu/2000index.html, (current Apr. 2008).

[11] P. Ning. TIAA: A Toolkit for Intrusion Alert Analysis. http://discovery.csc.ncsu.edu/software/correlator/, (current Mar. 2008).

[12] The MathWorks. MATLAB: The Language of Technical Computing. http://www.mathworks.com/, (current Dec. 2008).

[13] A. Faour, P. Leray, and B. Eter. Automated Filtering of Network Intrusion Detection Alerts. *Proc. 1st Joint Conf. on Security in Network Architectures and Security of Information Systems*. 2006, pp. 277-291.